NAVAL

POSTGRADUATE

SCHOOL

MONTEREY, CALIFORNIA

THESIS

A REALISTIC MODEL OF NETWORK SURVIVABILITY

by

Ozlem Ozkok

September 2003

Thesis Advisors: Geoffrey Xie
                                                   Alex Bordetsky

Approved for public release: distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY (*Leave blank*)	**2. REPORT DATE** September 2003	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: A Realistic Model of Network Survivability			**5. FUNDING NUMBERS**
6. AUTHOR(S) Ozlem Ozkok			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release: distribution is unlimited			**12b. DISTRIBUTION CODE**
13. ABSTRACT (maximum 200 words) This thesis focuses on evaluating network survivability and Quality of Service (QoS) in a network. There have been studies on developing network survivability metrics; however, the implementation of these survivability measures usually are based on unrealistic assumptions. This thesis has some experiment results based on identifying all min-cuts of a network and computing survivability of the nodes based on these criteria. The main contribution of the thesis is a novel approach to handling correlated or dependent component failures. In a complex network, it is not trivial to compute the probability of failures of the nodes even if the component failures are independent. With this new approach, network administrators could predict the optimal nodes in a network under more realistic conditions. Java-based simulation programs are developed to evaluate the approach. This project is motivated by network security problems in which a decision maker has to select nodes to host critical information servers when there is an attack to the network. The solution will give the decision makers criteria that would help them to make better decisions.			
14. SUBJECT TERMS Network Survivability, Network Attacks, Max Flow, Min-Cut, Probabilistic Networks, Modeling Dependent Nodes, Graph Algorithms			**15. NUMBER OF PAGES** 63
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

THIS PAGE INTENTIONALLY LEFT BLANK

A REALISTIC MODEL OF NETWORK SURVIVABILITY

Ozlem Ozkok
Lieutenant Junior Grade, Turkish Navy
E.E., Turkish Naval Academy, 1997

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

AND

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2003

Author: Ozlem Ozkok

Approved by: Geoffrey Xie
                 Thesis Advisor

                 Alex Bordetsky
                 Thesis Advisor

                 Dan C. Boger
                 Chairman, Department of Information Sciences

                 Peter Denning
                 Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis focuses on evaluating network survivability and Quality of Service (QoS) in a network. There have been studies on developing network survivability metrics; however, the implementation of these survivability measures usually is based on unrealistic assumptions. This thesis has some experiment results based on identifying all min-cuts of a network and computing survivability of the nodes based on these criteria.

The main contribution of the thesis is a novel approach to handling correlated or dependent component failures. In a complex network, it is not trivial to compute the probability of failures of the nodes even if the component failures are independent. With this new approach, network administrators could predict the optimal nodes in a network under more realistic conditions. Java-based simulation programs are developed to evaluate the approach. This project is motivated by network security problems in which a decision maker has to select nodes to host critical information servers when there is an attack to the network. The solution will give the decision makers criteria that would help them to make better decisions.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to acknowledge Turkish Navy and Turkish Republic for giving me such a good educational opportunity.

I would like to thank to my Mom, Guler Ortunc for the support she gave me throughout my life. I would never be where I am if she did not stand by me my entire life.

Most special thanks to my husband, Murat Ozkok. The person who tries to understands me most. He is my best friend. I can never forget all we've done for each other, or all we've been through together.

I would also like to thank my advisors Prof. Geoffrey Xie and Prof. Alex Bordetsky for their time, guidance and understanding during the whole research process.

Finally, I would like to dedicate this thesis to my beloved Mom, Guler Ortunc.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION AND OVERVIEW

A. BACKGROUND

Survivability is typically defined along the following lines: the ability to continue to fulfill a mission even in the face of attacks and failures. The critical thing in this definition is that it is impossible to stop all attacks and prevent all failures. "No single component of a system is immune to failure or subversion."[DIE01]

Network Survivability is critical to Network Centric Warfare. It is also an important concern of homeland security because computers are part of the national critical infrastructures and must have high survivability while facing terrorist attacks and natural disasters.

The growth of the Internet has produced the emergence of a global information society. Businesses can function internationally with great efficiency exchanging information seamlessly across their supply chains. Governmental use of the Internet will increasingly extend to international information sharing and collaboration. Perhaps the greatest threat to the Internet is the security of so many systems connected to it. A lack of security expertise by most of the Internet users results in vulnerabilities in the network that can be compromised by motivated attackers.

There has been some studies on developing network survivability metrics, however the implementation of these survivability measures usually require money, big design changes to protocols and systems. As stated in [XIE02] Ellison provided a general definition of network survivability in [ELL99] and described some solution approaches to the problem. Another paper by the same authors defined a software engineering process for designing survivability into application [MEA00]. Furthermore, some concrete results are presented in some contemporary papers, e.g., [SUL99], [UMA01], [WEL00]. Nevertheless, their focus was still on how to make software agile to effectively detect and react to system component failures and software errors. In [WEL00], a customizable utility function is used to indirectly measure survivability of a system configuration from the point of view of the system user. Jha and Wing proposed a formal framework based on Bayesian networks for reasoning about the survivability properties of distributed

systems [JHA00]. The work was rigorous but the proposed algorithm was too complex for large networks. Unfortunately, the studies that have been done on network survivability so far are not mature enough and they lack quantifiable metrics.

To address this lack of a network survivability measure, a global connectivity metric was developed in the thesis of another graduate student, Baris Aktop [AKT03]. This thesis contributed to the thesis work of Eng Hong Chua [CHU03], who developed a heuristic for comparing the connection reliability of two nodes to a common destination node when these two nodes have the same number of edge-disjoint paths to that destination. The heuristic is based on estimating the probability of each of the nodes being cut from the server given same number of link failures. This thesis includes some experimental results based on identifying all min-cuts of a network and computing survivability of the nodes based on these criteria.

B.   SCOPE

In today's competitive and dynamic information technology environment, there is a need for IT security as an integral component of the IT architecture of enterprises. The concept of "survivability metrics" and "security metrics" including test, evaluation, criteria identification, quantification of strengths, risk assessment/analysis and other related activities have been explored since 1995. However, these efforts have provided neither generally accepted nor reliable measures for rating information systems' survivability and security. Moreover, inconsistent terminology has complicated the development of IT metrics, such as rating, ranking, quantifying or scoring measurements.

There are three questions that should be asked when quantifying network survivability:

1.   WHAT you need to measure (e.g. technical, system)

2.   WHY you need to measure (e.g. comparison, description)

3.   WHOM you are measuring for (e.g. Technical experts, decision makers)

This thesis is motivated by network security problems in which a decision maker has to select between nodes to host critical information services when there is an attack to

the network. The goal of this thesis is to give the network administrators criteria that would help them to make better decisions. These criteria can be used to develop heuristics and perform network reliability analyses to understand and better protect the networks.

The main contribution of the thesis is a novel approach to handling correlated or dependent component failures. In a complex network, it is not trivial to compute the probability of failures of the nodes even if the component failures are independent. With this new approach, network administrators could predict the optimal nodes in a network under more realistic conditions. Java-based simulation programs are developed to evaluate the approach.

C.   OVERVIEW

This thesis is structured into the following chapters:

Chapter II: Network Survivability Concept. Describe the concept of survivability. Discuss the importance of network survivability in Network Centric Warfare.

Chapter III: Connectivity Based Survivability Metric. Define the basic terms of the graph theory; describe the connectivity based survivability metric. Describe the computation of the Ke metric and minimum number of cuts in a network. Introduce another algorithm to evaluate ranking of the nodes in a network.

Chapter IV: A Heuristic Model for Determining the Survivability of the Connection. Describe a heuristic model, called Pe, and discuss the assumptions of the model. Validate and refine the heuristic model.

Chapter V: A Realistic Approach for Network Survivability. Introduce an analytical model and algorithm to evaluate network survivability under more realistic conditions.

Chapter VI: Conclusions and Future Work. Summarize the results from the thesis and recommend future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II.   NETWORK SURVIVABILITY CONCEPT

A.   INTRODUCTION

There has been a big improvement in network services as a consequence of developments in networking technology and the Internet. Government agencies and businesses are increasingly dependent on networked systems. The security of these systems remains a big problem because of the transparency of the public networks. Hardening of the information systems is never enough unless the systems are physically isolated. As long as the networks stay connected, there is going to be people who would like to attack them for various purposes.

Survivability is the capability of a system to fulfill its mission on time while attacks, failures, or accidents are present [FIS99]. The term, mission, refers to high-level organizational objectives and mission fulfillment can be evaluated by the results achieved by the system in the context of operational conditions.

While robustness, normally associated with fault tolerance in networks has long been an issue in providing service assurances in the presence of component failures, survivability is a new concept in non military networks. It ensures that a system can continue to deliver essential services even in the presence of attacks [CHU03]. Current network architectures, such as that of the Internet, rely on sophisticated, stand-alone routers. They are being overwhelmed with the introduction of the management functions while coming under more aggressive threats.

In an integrated services network, quality of service (QoS) levels to individual user sessions must be guaranteed. To ensure QoS, the network has to reserve resources for a set of packets at particular routers. Additionally, an integrated services network must support real-time applications that have stringent packet delay requirements [XIE98].

This thesis focuses on the failures and survivability of mission critical servers that deliver different network functions, such as resource management, routing, accounting, network management and security. Such servers might include DHCP, DNS, or domain controllers, among others

In summary, a heavy weight node in a network can be a performance bottleneck. Therefore, responsiveness, scalability, and fault-tolerance are major concerns in a network design from the survivability point of view.

B.   CONCEPT OF SURVIVABILITY

The survivability concept covers a broad range of engineering areas, such as security, fault-tolerance and reliability. Survivability research builds upon reliability research to focus on recovery after a failure in the system while reliability research assumes that failures can happen but that mission critical functions of the system must be active despite the failures [YUR99]. One of the best examples of survivability research is combat aircraft that can still fly even though they have experienced some extensive system damage.

In the computer/ telecommunications infrastructure some part of a network is always down due to attacks or failures. Network managers test and collect data to understand why systems fail and to determine why some systems fail less or more than others. This kind of reliability analyses assumes that failure events are independent (for mathematical analysis). In this thesis, our goal is to compute the probability of failures of the nodes, including dependent/conditional node failures given the dependencies and probability of failure statistics.

Although the security and survivability approaches are different, they cannot be separated. A security approach in a network tries to identify the holes, or vulnerabilities, of the systems and harden them. Survivability ensures that a network performs its functions in the midst of attacks or failures, while security ensures high system resistance to attacks. For example, safes are traditionally classified according to how long they can be expected to resist certain types of attacks, such as break-ins. In this example, survivability of the safe is a decision criterion from the security point of view. Therefore, when we analyze networks, we need quantifiable measures from both the security and survivability points of view. [DIE01].

The design and evaluation of a survivable system requires consideration of reliability and security, adaptability, efficiency, and cost-effectiveness. Nowadays, markets tend to focus on minimalist solutions: just-in-time etc. As a result, systems tend to be meta-stable and they eventually collapse. Robust solutions require more expensive details. It is clear that the design and evaluation of survivable systems is hard. Even the question of defining appropriate metrics is difficult. The author suggests that the more appropriate approach is to focus on more realistic examples. This thesis defines dependent node failures in a network and provides a method to compute them without relying on tight assumptions.

C. NETWORK CENTRIC WARFARE AND HOMELAND SECURITY

As computer technology has become increasingly integrated into modern military organizations, military planners have come to see it as both a target and a weapon. Countries are developing and implementing cyber strategies designed to impact an enemy's command and control structure, logistics, transportation, and other critical functions. As a RAND corporation study pointed out in the mid-1990s, the entry costs for conducting cyber war are extremely modest [SHI01].

In a limited cyber war, the information infrastructure is the medium, target and weapon of attack, with little or no real-world action associated with the attack. An insider might place malicious software directly within the enemy's network. Degrading the level of service of the network may cause the enemy to use alternate routes, which may cause additional vulnerabilities. Denial-of-service attacks would require different approaches when there is no Internet access in the systems that are supporting critical, national infrastructures. A failure of emergency services in major cities would not only result in many people dying, but also would make people lose confidence in government, thereby generating both physical and psychological effects.

Network Centric Warfare (NCW) is based upon the experiences of organizations that have successfully adapted to the changing nature of their competitive spaces in the Information Age. The centrality of the information and its potential as a source of power is the source of the power of NCW. NCW gives a new framework for analyzing military

7

missions, organizations, and operations. Figure 1 shows the Military as a Network-Centric Enterprise [http://www.dodccrp.org/NCW/ncw_chapter.htm, September 2003] It shows the infrastructure that is expected to enable shared battle space awareness and knowledge. The NCW framework will increase the tempo of operations, responsiveness, and combat effectiveness. At the same time, it will lower risks and costs.
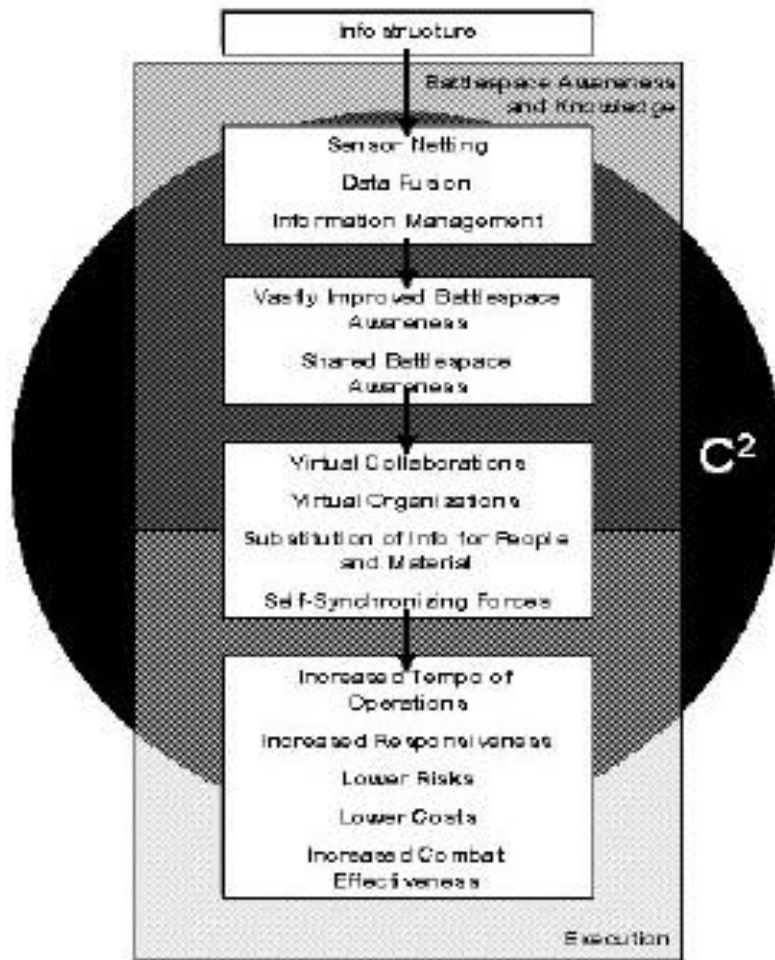
Figure 1.   Military as a Network-Centric Enterprise

NCW is built around the concept of sharing information and assets. This is enabled by networking battle space entities together. In NCW, capabilities for sensing, commanding, controlling, and engaging are robustly networked. The source of increased power in a network comes from the content, quality, and the timeliness of the information flowing in the network. The structural or logical model of the NCW is given in Figure 2[CEB98]. There is a high-performance information grid that enables the operational architectures of sensor grids and engagements grids. Sensor grids generate high levels of

battle space awareness quickly and synchronize awareness with military operations. Engagement grids translate the awareness into increased combat power. The cooperative engagement capability (CEC) combines a high-performance sensor grid with a high-performance engagement grid. (See Figure 3)
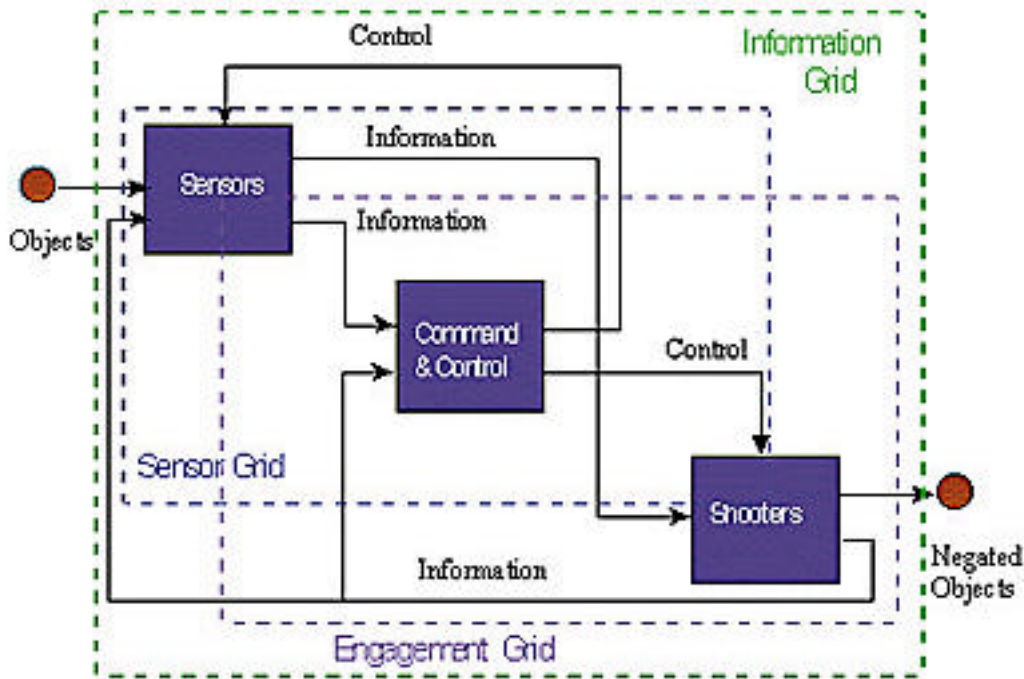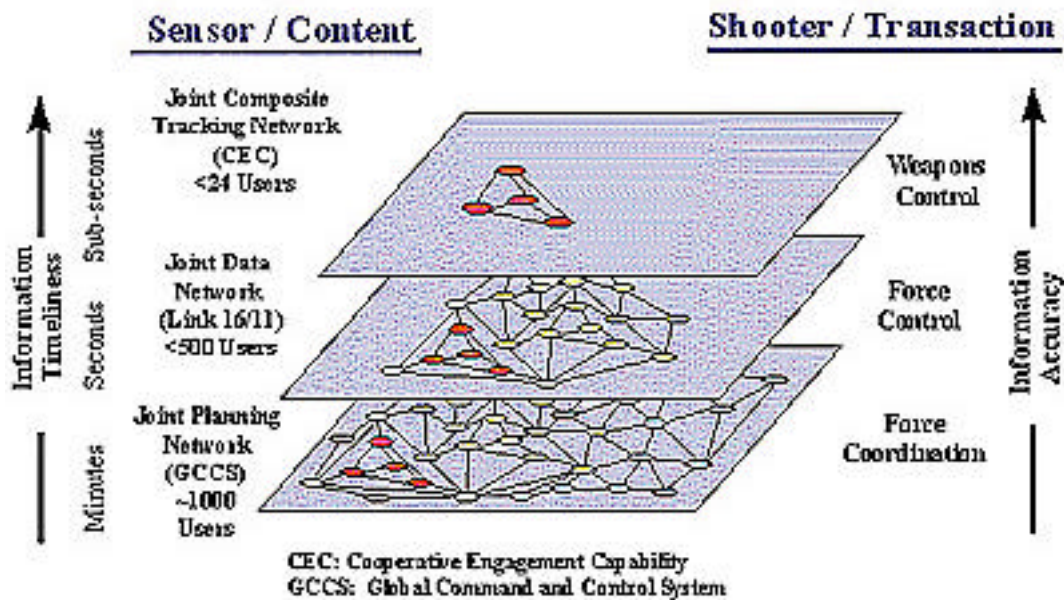

Figure 2.   Architecture for NCW

Figure 3.   Cooperative Engagement Capability (CEC)

9

Virtual collaboration in the information domain has many operational benefits. In the following three examples, these benefits are highlighted.

Example 1: New Relationships Between Commanders—Battle Command via VTC

Old Way: Corps and division commanders travel across the battlefield to be in the same place at the same time to plan ground operations.

Network Centric Warfare: Commanders interact via VTC, which results in a significant reduction in planning time and elimination of travel time.

Value: Decreased planning time provides commanders with the operational flexibility to enable their forces to rehearse, move-to-contact, re-supply, repair, or rest. The net result is increased combat power.

Concept Status: Demonstrated by U.S. Army in operational exercises.

Example 2: Quality of Life

Old Way: Deployed forces communicate with families and loved ones via mail or telephone, at infrequent intervals.

Network Centric Operations: Deployed forces communicate with families and loved ones with increased frequency and timeliness via e-mail (potentially on a daily basis), telephone, or VTC.

Value: Deployed war fighters are able to solve family problems in close to real time (e.g., finance), interact with their children, and experience their children's lives while they are growing up. Worry goes down, morale goes up, and operational effectiveness remains at a higher level over long deployments. Although, operational security must be closely monitored and enforced to ensure missions are not compromised.

Concept Status: Operational.

Example 3: Distance Learning

Old Way: Units release war fighters to attend training or education events away from their duty locations.

Network-Centric Operations: Education is provided to warfighters deployed with their units via VTC or compact disk (CD-ROM).

Value: Manning levels are maintained and opportunities for education and training are available to all deployed forces. Operational proficiency and morale increase.

Concept Status: Operational [http://www.dodccrp.org/NCW/ncw_chapter.htm, September 2003].

For Network Centric warfare to work, the right data must be available to the right people at the right time. For example, satellite imagery of a threat from a mobile Scud launcher is important information. It needs to be accessible by the key planners and attack pilots [LAW00].

There are many examples of cyber terrorists' attacks recently. One of the stories about these attacks focused on the Massachusetts Water Resource Authority (MWRA), the agency that controls water for much of eastern Massachusetts [DES02]. A cyber intruder can easily exploit the computers that control the flow of water. However, even if a hacker penetrates its network, the MWRA has a multitude of checks that ensures contaminated water never reaches people. Preventing physical harm caused by a cyber attack is easier than protecting valuable data from cyber attacks. Experts agree that the most harmful cyber attack threat is the one that combines these two intended results.

One expert, Mark Fabro, president and chief scientist at Terrosec Corporation, a security consulting firm in Toronto, says it might be possible to identify not only the principle components of the network that controls the national power grid, but also the physical location of these components. In that fashion, a cyber terrorist would either know which network components to attack or where the most exposed vulnerability exists for physical attacks. "That kind of information, combined together, could be used to devastate elements of the critical infrastructure," Fabro says.

The National Infrastructure Protection Center, an organization charged with protecting critical U.S. infrastructure, in January 2003 issued a bulletin warning that a computer, owned by an individual with ties to Osama bin Laden, contained information about the structural engineering of dams and other water-retaining structures [DES02]. The bulletin said law enforcement agencies had "received indications" that other Al-Qaeda members were interested in water supply and waste management practices and were culling information about insecticides and pest control practices from several Web sites. (See: http://www.nipc.gov/publications/infobulletins/2002/ib02-001.htm, September 2003) The government is making a concerted effort to ensure that its own Web sites don't offer any assistance to terrorists. On March 19, the Bush administration went so far as to order all government agencies to remove from public view any information on "weapons of mass destruction, as well as other information that could be misused to harm the security of our nation and the safety of our people."[DES02].

There are other examples of attacks against information systems. In south Florida, a hacker was able to break into local government systems and divert 911 calls to a local pizza parlor. In Houston, Texas, FBI officials caught a hacker before he could insert a worm into computers that would have resulted in the widespread shutdown of 911. And in 1997, a young hacker shut down communications at an FAA tower in Worcester, Massachusetts, for six hours. These attacks are not limited to the continental United States, as NATO servers were shut down for several days during the 2000 bombing campaign in Serbia and Kosovo.

These examples underscore the necessity that network designs integrate notions of robustness and survivability in the hosting of critical missions. At the same time, contingency plans are required for the recovery of critical roles. Therefore, a solution for networks that will make them operate efficiently and safely is proposed

III.   CONNECTIVITY BASED SURVIVABILITY METRIC

A.   INTRODUCTION

In this thesis the author focused on a connectivity-based survivability metric developed in [AKT03] and evaluated the heuristic developed in [CHU03], which is used for comparing the connection reliability of two nodes to a common destination node when these two nodes have the same number of edge-disjoint paths to that destination. The heuristic is based on estimating the probability of each of the nodes being isolated, or cut, from the server given some number of link failures. This thesis provides some experimental results based on identifying all min-cuts of a network and computing survivability of the nodes based on these criteria.

Two criteria were used to compare and rank the nodes of a network:

1.   Network survivability metric based on the edge-connectivity factor (K_e).

2.   Probability of failure of a link given K_e number of edge failures.

These criteria are investigated for a given node collection in sequence. That is, if two nodes have the same value for K_e, then the second criteria is examined.

During the research, this heuristic was validated and refined. Given K_e number of failures, the nodes in a network can always be ranked, as mentioned in Section C of this chapter. The definitions and formulas used to calculate this connectivity-based survivability metric are provided below.

B.   DEFINITIONS OF THE TERMS USED IN GRAPHS AND ALGORITHMS

A network is modeled using a graph consisting of nodes representing communications centers and edges representing the links between communication centers. A graph G, which is denoted by (V, E) consists of a set of nodes or vertices, V, and a set of edges, E. Each element of E is an unordered pair (v_i, v_j), where v_i and v_j are elements of V.

A graph is called undirected graph if it consists of undirected edges. A loop is a set of one or more sequential edges that originates and terminates at the same node.

A Path is a walk in which all edges and all vertices on the walk are unique, except that the first and last node may be the same. Edge-disjoint paths are paths with no edges in common. Node-disjoint paths are paths that share no common nodes other than the source and destination nodes.

A cut-set is a set of edges whose removal disconnects the graph. A minimum cut-set, or min-cut, is a cut-set that contains the fewest possible number of edges which when removed disconnects the graph.

C.   CONNECTIVITY BASED SURVIVABILITY METRIC

To find the most optimal location (node) for the server in a network, the clients must be offered reliable connectivity to the server. The probability that a client will survive a number of edge failures is dependent on the order of the edge-disjoint paths between server and the client. The greater the number of edge-disjoint paths between the server and the client, the better and more reliable the connection is. In order for the path between a node and a server to be non operational, there must be at least as many edge failures as there are edge-disjoint paths (K_e) between the two.

For example, if an assessment of the robustness of the connectivity between two nodes and a third node is going to be done, looking at the K_e can be a good start to the decision process. However, the K_e of the two nodes might be equal. In that case, finer granularity in computing the connectivity of the nodes is necessary. The second criteria, the probability of a node being disconnected given K_e number of edge failures, should then be examined. This probability is given in Equation (1) [XIE02].

$$P_r\{cut(s,d)=1\} = \sum_{i=K_e(s,d)}^{E} P_r\{cut(s,d)=1\,|\,i\,edge\,failure\}P_r\{i\,edge\,failure\} \quad (1)$$

When two nodes, for example s_1 and s_2, have the equal K_e values, the comparison is done by $P_r\{cut(s_1,d)=1\} = |\,K_e\,edge\,failure\}$ and

14

$P_r\{\mathrm{cut}(s_2,d)=1\}=|K_e$ edgefailure} values. The location with the smaller probability has a higher survivability.

D.   COMPUTATION OF K_E

In a network, the maximum flow value, when unit weights are assigned to all the edges in the graph, is equal to the K_e value. There are different algorithms to compute the maximum flow in a network. An open-source, Java-based algorithm platform (JGAP) was downloaded from http://im.ncnu.edu.tw/~tsai/definite/JGAP/ JGAP.html, September 2003. Finding Ke by Ford-Fulkerson's maximum flow algorithm was implemented in Java by Baris Aktop [AKT03]. The pseudo code for this algorithm is given in Figure 4.

```
        procedure MaximumFlow(s, d)
        begin
                // Part I: Setup
    1.          start with null flow: f(u,v) ← 0 ∀(u,v) ∈ E;
    2.          initialize c(u,v) ← edge capacity (weight) of edge(u,v) ∈ E;
    3.          initialize residual graph to be original graph;

                // Part II: Loop
    4.          repeat
    5.              search for augmenting path from s to d in residual graph;
    6.              if (path p found) then
    7.                  c_f(p) ← min{c(u,v)|(u,v) ∈ p};
    8.                  for (each (u,v) ∈ p) do
    9.                      f(u,v) ← f(u,v) + c_f(p)
    10.                     remove (u,v) from residual graph;
    11.         until (no augmenting path);
    12.         initialize flow = 0;
    13.         for (each (u,v) ∈ E) do
    14.             flow = flow + f(u,v);
    15.         return flow;
        end
```

Figure 4.   Pseudo code for finding the maximum flow

The complexity of this maximum-flow algorithm is O $(N\,|E|^2)$, where N is the number of vertices and E is the total number of edges in the graph. The algorithm uses

15

augmenting paths to find a path of positive capacity from the source node to the destination node and adds it to the flow. This adding continues till no more augmenting paths are found in the graph. The output of the algorithm is the *flow* variable, which contains the summation of flow capacities of all augmenting paths. Breadth-First Search (BFS) is used in the maximum flow algorithm to find augmenting paths because it ensures the paths chosen are of minimum length.

E.   COMPUTATION OF MINIMUM CUTS (MIN-CUTS)

As noted, if the K_e values of nodes under consideration are equal, looking at the conditional probability of connectivity failures for the nodes, given K_e, might be a second criterion to break the tie. To compute this probability, it is necessary to find out the number of min-cuts between the server and the client.

There are critical edges that have no alternate paths between a source and the destination nodes. Removal of these critical edges will cause the path between source and destination to be disconnected. The more critical edges that exist in the path to a node being considered the lower the survivability of the connectedness of that node. The number of critical edges determines the number of min-cuts in a graph. The pseudo code for the algorithm to enumerate them is given in Figure 5. Currently, there is no known algorithm that is able to solve all types of graphs in polynomial time. The complexity of the algorithm is $O(E^{Ke})$.

```
        procedure FindMinCuts()
        begin
1.              count = 0 ;
2.              numOfCutSets = 0 ;
3.          while count != ^EC_{K_e} do
4.              select K_e edges;
5.              G' ← remove selected K_e edges from G ;
6.              maximum flow algorithm to find K_e';
7.              if K_e' = K_e then
8.                  numOfCutSets ← numOfCutSets +1 ;
        End
```

Figure 5.   Pseudo code for finding the min-cuts

Figure 6 shows an example graph. The source is Node 0 and the destination is Node 5. Figure 7 shows the edge-disjoint paths in different colors. The K_e, which is the edge-connectivity of Node 0, is 2. Figure 8 shows in how many milliseconds the algorithm was able to identify the min-cuts.

Enumerating all min-cuts in a graph is an NP-hard problem and is not likely to get solved in polynomial time, depending on the graph topology. The proposed algorithms were able to identify all min-cut sets in polynomial time for certain type of graphs.

In Figure 9, there are 4 edge-disjoint paths between Node 0 and Node 5. The K_e value for the Node 0 is 2 and 16 min-cuts were enumerated by the algorithm in 300ms, as shown in Figure 11.
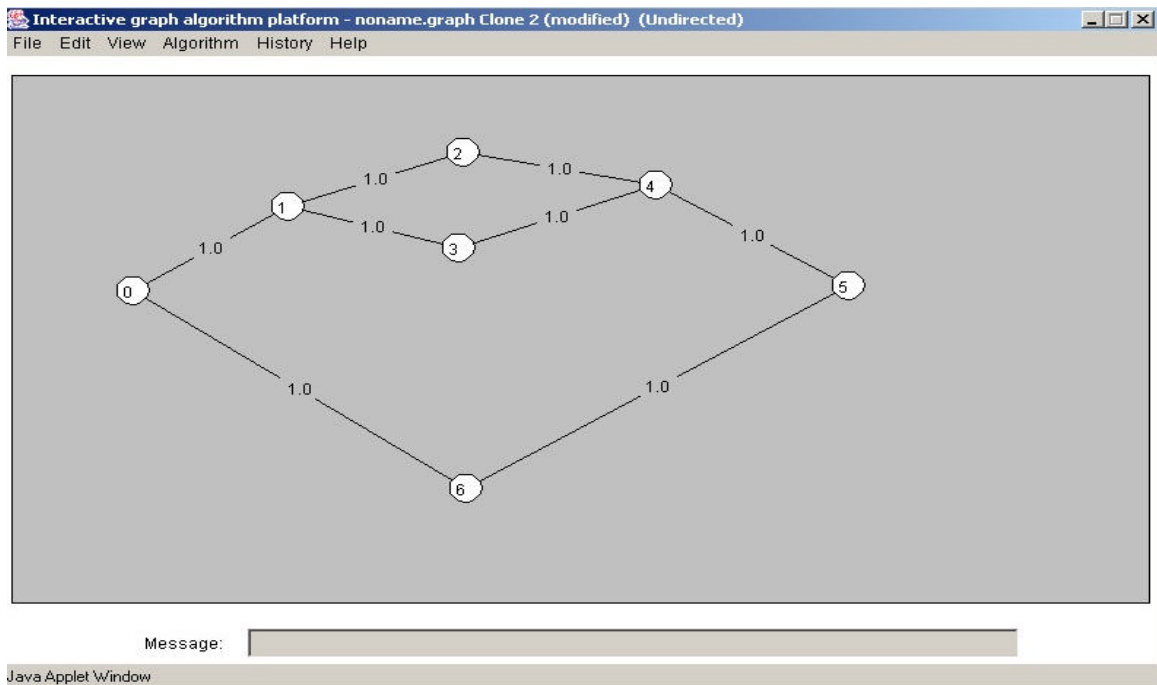
Figure 6.   Example Topology-1

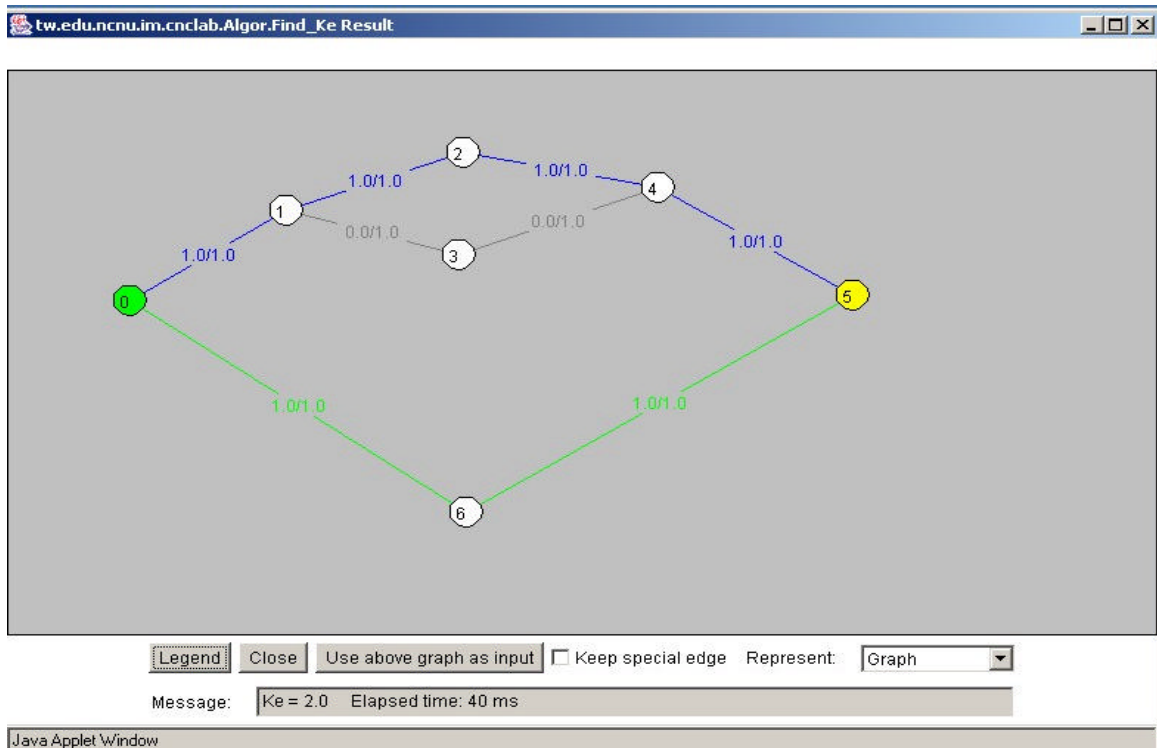Figure 7.   Edge-disjoint paths and Ke value of source node 0

Figure 8. Number of mincuts found in 120ms.

The number of min-cuts in Figure 8 is identified as follows:

There are 4 Min-cut sets : (0 , 6) (0 , 7) (5 , 6) (5 , 7)

Edge 0 is between Vertex 0 and Vertex 1

Edge 5 is between Vertex 4 and Vertex 5

Edge 6 is between Vertex 0 and Vertex 6

Edge 7 is between Vertex 6 and Vertex 5

Figure 9.   Example Topology-2
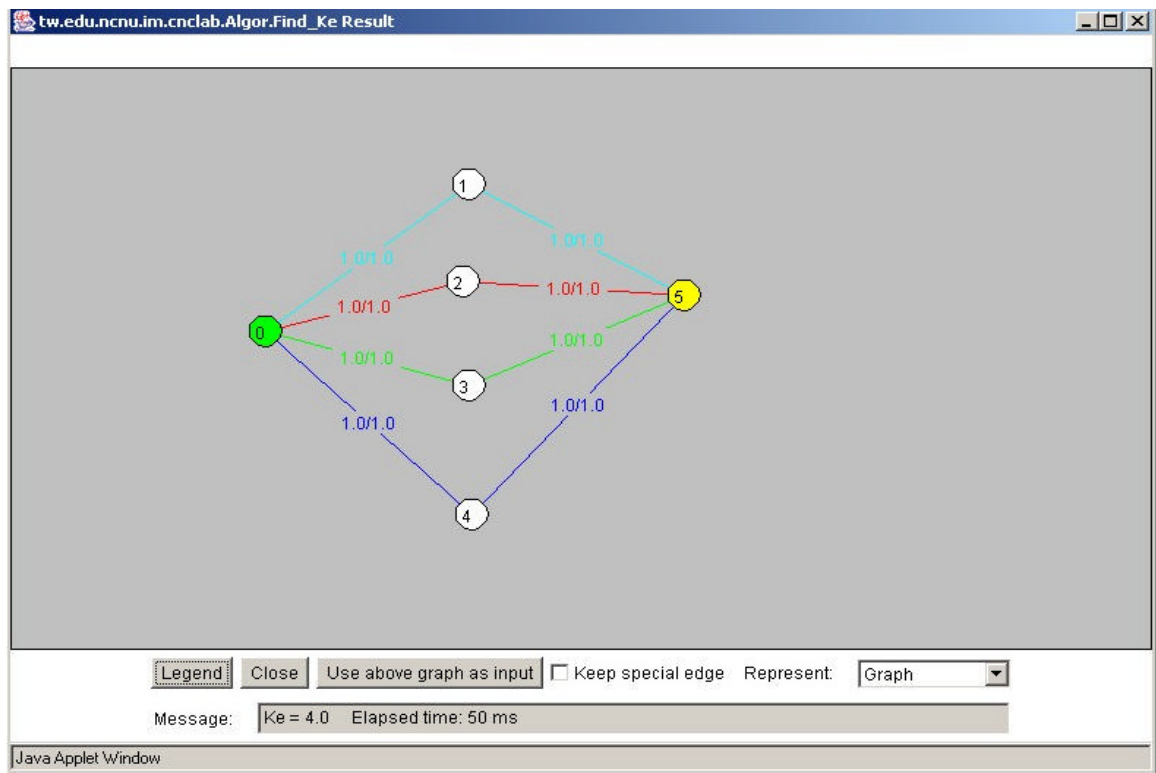
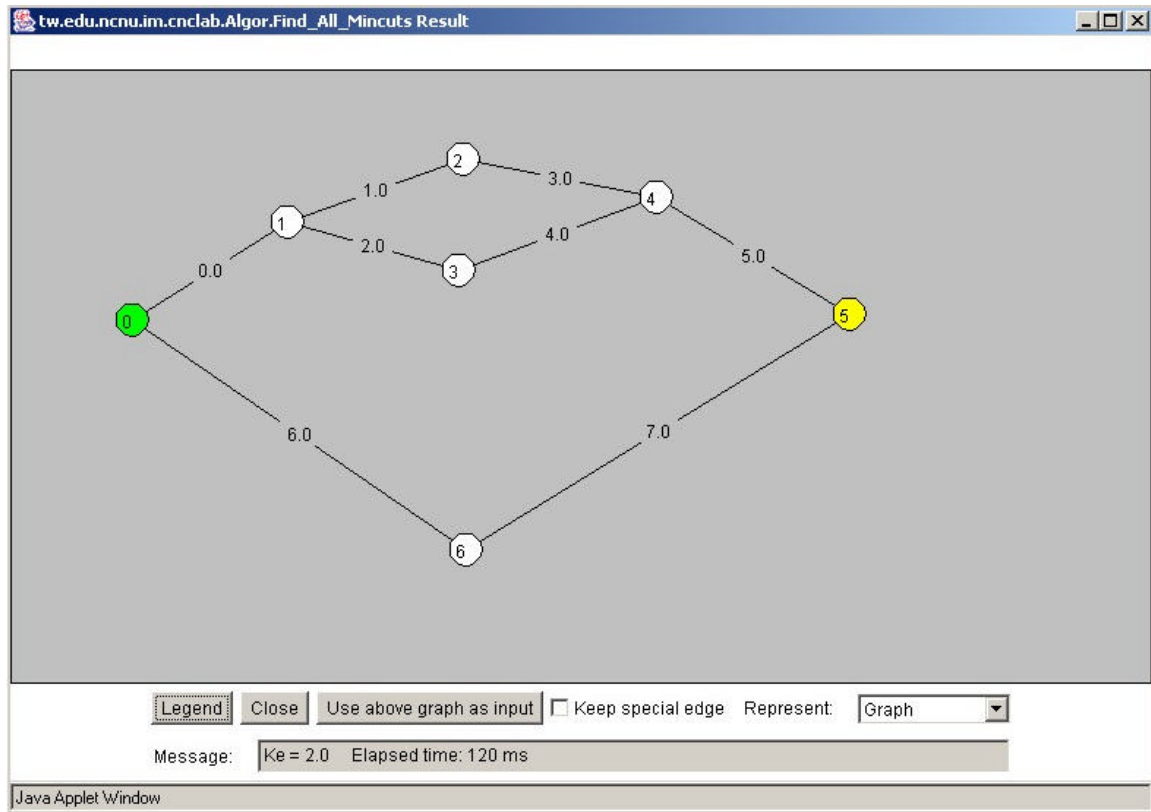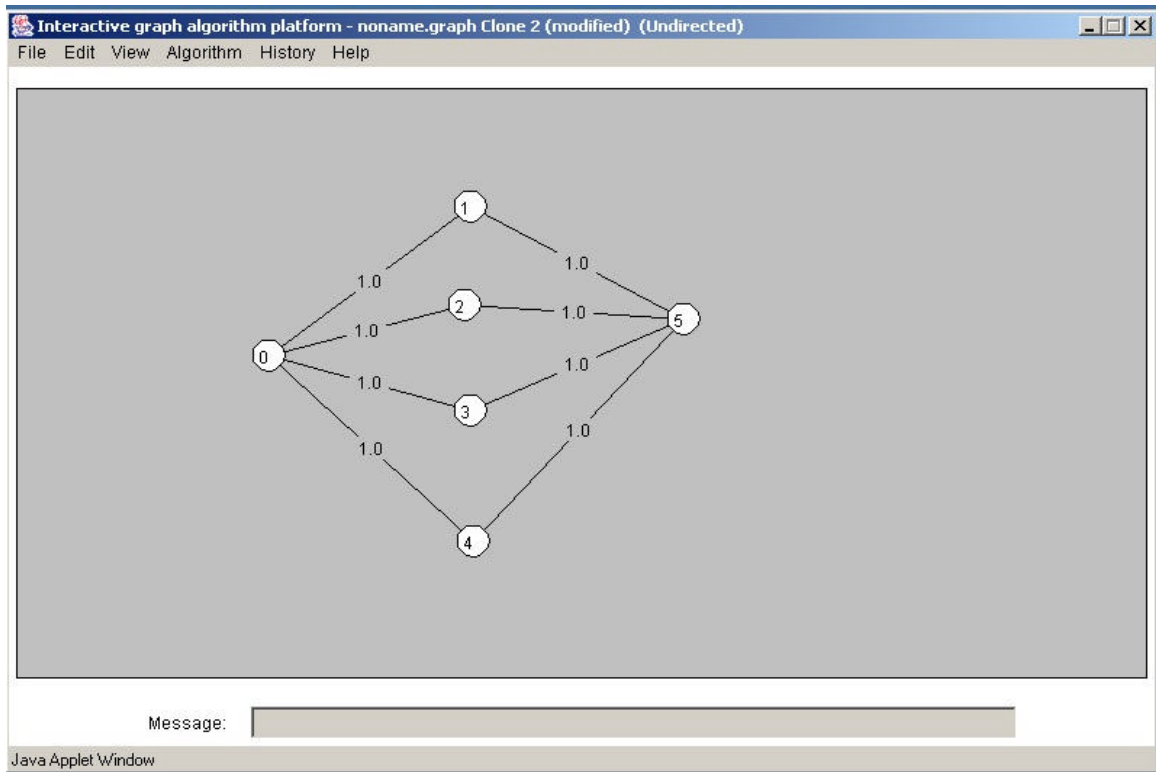Figure 10.   Showing 4 edge-disjoint paths between 0 and 5
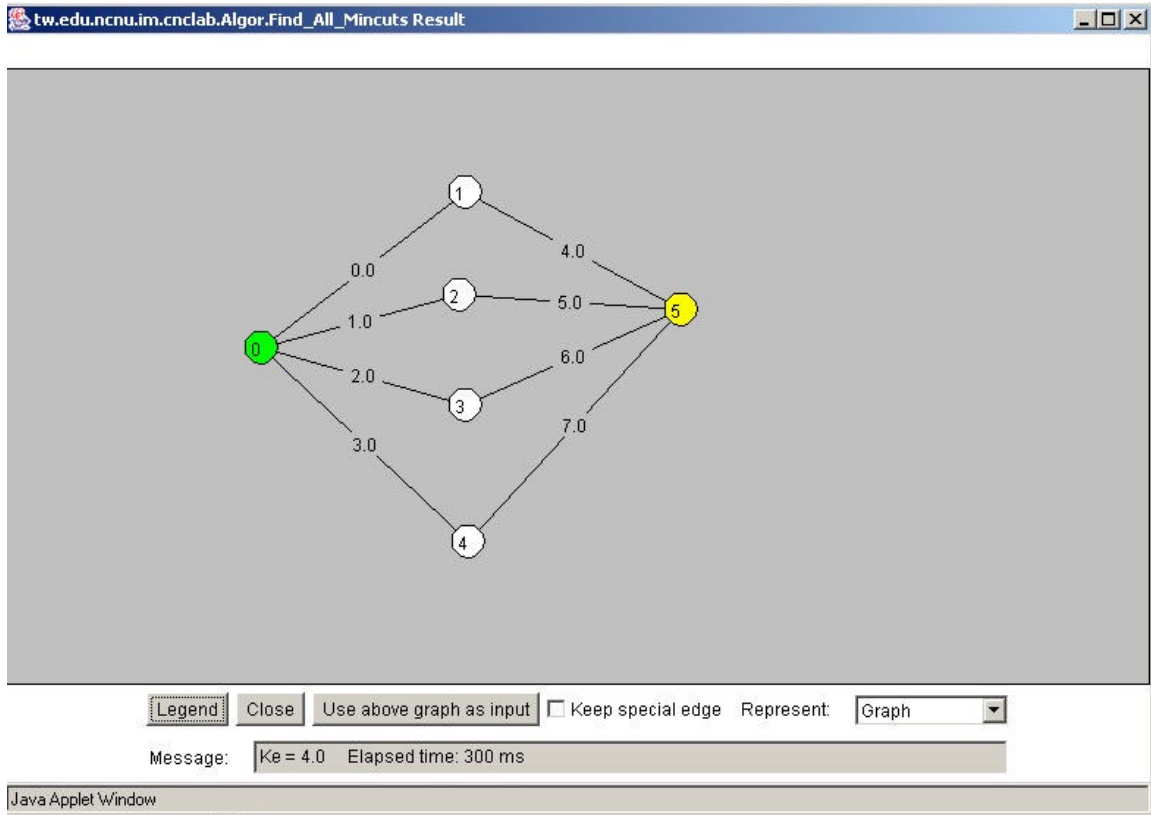
Figure 11.   Number of Mincuts found in 300ms.

The number of min-cuts in Figure 11 is identified, as follows:

There are 16 Min-cut sets : (0 , 1 , 2 , 3) (0 , 1 , 2 , 7) (0 , 1 , 3 , 6) (0 , 1 , 6 , 7) (0 , 2 , 3 , 5) (0 , 2 , 5 , 7) (0 , 3 , 5 , 6) (0 , 5 , 6 , 7) (1 , 2 , 3 , 4) (1 , 2 , 4 , 7) (1 , 3 , 4 , 6) (1 , 4 , 6 , 7) (2 , 3 , 4 , 5) (2 , 4 , 5 , 7) (3 , 4 , 5 , 6) (4 , 5 , 6 , 7)

Edge 0 is between Vertex 0 and Vertex 1

Edge 1 is between Vertex 0 and Vertex 2

Edge 2 is between Vertex 0 and Vertex 3

Edge 3 is between Vertex 0 and Vertex 4

Edge 4 is between Vertex 1 and Vertex 5

Edge 5 is between Vertex 2 and Vertex 5

Edge 6 is between Vertex 3 and Vertex 5

Edge 7 is between Vertex 4 and Vertex 5

In graphs like that in Figure 11, when the numbers of nodes on the paths increase and the number of edge-disjoint paths increases, assuming no cross-paths between them, as shown in the topology in Figure 12, the total number of min-cuts is n^{Ke}, where (n-1) is the number of nodes that exist between the source and destination on an edge-disjoint path. This makes the time to enumerate all min-cuts grow exponentially with the number of K_e.

F.   ANOTHER ALGORITHM FOR ENUMERATING THE ALL MINIMUM WEIGHT AND NEAR-MINIMUM S-T CUTS

Because of the complexity of the algorithm given in Figure 5, the Java program given in [AKT03] was not suitable for large network topologies. Therefore, a second Java implementation for enumerating all min-cuts in a given graph was used to run the simulations. The Java code is detailed in [WOO00].

Briefly, this enumeration algorithm is based on a recursive "inclusion-exclusion" method. The algorithm identifies a min-cut by finding the maximum flow, using Ford-Fulkerson's maximum flow algorithm, in the network and then partitions the space of minimal cuts by attempting to include and exclude specific edges. In this algorithm, if the network edges have different integer weights assigned, by setting the variable e greater than zero, near-minimum weight s-t cuts can be found, too. A cut is a "near-minimum" if its weight is less than the product of (1+e) and the minimum cut weight, for some e = 0.

The complexity of the algorithm for finding only minimum cuts (when e = 0) is O (f ($|V|$, $|E|$) + $|V||E||C_{0\ (G)}|$) where f ($|V|$, $|E|$) is the complexity of solving a maximum flow problem on G = (V, E) and $C_{0\ (G)}$ is the set of minimum cuts in graph G. The worst-case complexity of the algorithm for near-minimum cut enumeration remains unknown when e > 0.

In this project, the topologies shown in Figures 6 and 9 were given as inputs to the Java program and simulation results were obtained in 80ms and 50ms respectively. It was observed that enumerating all min-cuts by this algorithm was more efficient than the one described in Section 5. Since all the graphs used as examples in this paper had edge-weights of one, near-minimum cut enumeration was not tested. More results can be found in [WOO00].

Neither of the Java implementations used to simulate the various types of graphs were able to complete running with 1 node, which is 2 links, on an edge-disjoint path between the source and destination where 40 edge-disjoint paths were discovered. This makes the number of min-cuts to evaluate 2^{40}, which is more the a trillion min-cuts.
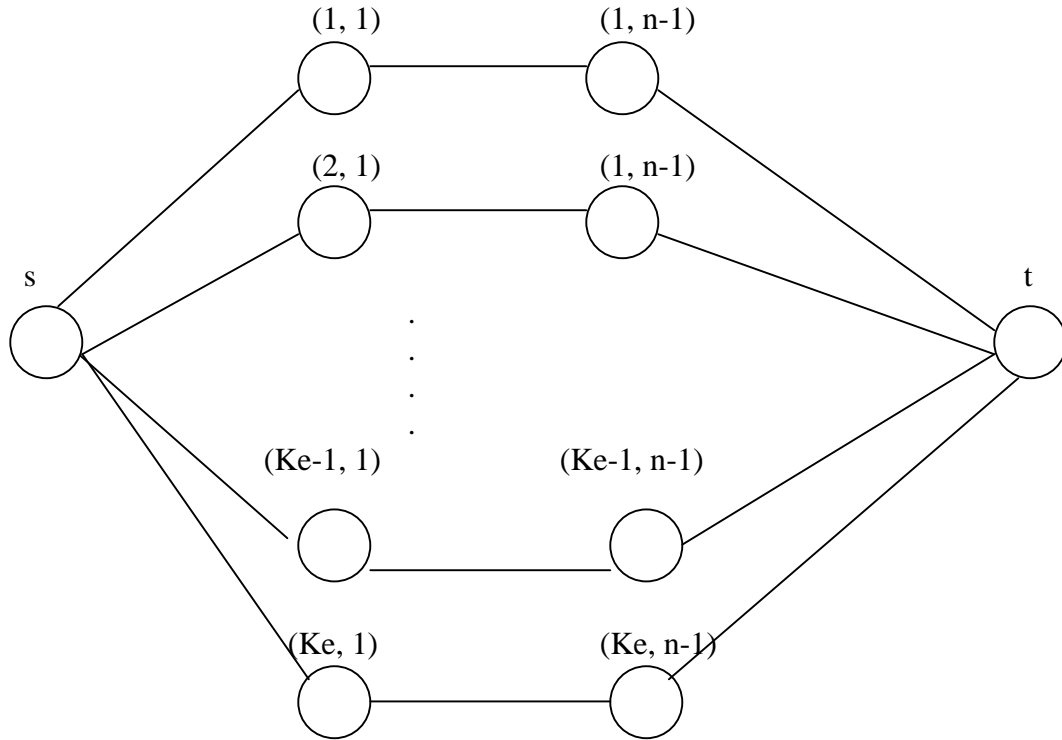

Figure 12.   Example topology – 3

THIS PAGE INTENTIONALLY LEFT BLANK

IV. A HEURISTIC MODEL FOR DETERMINING THE SURVIVABILITY OF THE CONNECTION

A. PE MODEL

The probability estimator model, P_e, determines the survivability of the connection based on the number of critical edges in each edge-disjoint path between the two nodes. An edge is critical if its failure will disconnect the path. So, the less critical edges, the lower the probability of path disconnection.

In the P_e model, the goal is to approximate $P_r\{\text{cut(s,d)} = 1 | K_e \text{ edge failures}\}$, the probability of connection failure given K_e number of edge failures. The connection between the two nodes fails if and only if the edge failures disconnect all the edge-disjoint paths. In other words, each of the edge failures must be a critical edge of a different edge-disjoint path for the connection to fail [CHU03].

The algorithm used to compute P_e is shown in Figure 13. The P_e model has as its core the essential idea of finding the number of critical edges, C_{P_i}, in Path i. In the algorithm for evaluating each augmenting path, Line 3 initializes the number of critical edges, C_{P_i}, to the length of the path, i. At line 4, each edge and consecutive sequence of edges are checked for an alternative path. If an alternate path is found then the sequence of edges between the ends of the alternate paths, vertices u and v, are not critical, therefore, C_{P_i} is decreased by the value equal to the distance between Vertex u and Vertex v. With the critical edges for each path known, P_e can be computed from the Equation (2).

$$\text{Pe} = K_e! \times \prod_{i=1}^{K_e} \left(\frac{C_{P_i}}{E - i + 1}\right) \qquad (2)$$

```
procedure FindPe(s, d)
begin
1.        G' ← remove edges of augmenting path from G;
2.        for each P_i ∈ P_{K_s} do
3.            C_{P_i} ← HopCount(P_i);
4.            initialize u ← s and v ← d;
5.            while u ≠ d do
6.                use Maximum flow algorithm to find an alternate path from
                  u to v in G';
7.                if alternate path exists then
8.                    C_{P_i} ← C_{P_i} − HopCount(u, v);
9.                    u ← v;
10.                   v ← d;
11.               else
12.                   v ← previous node of d on P_i;
13.                   if v = u then
14.                       u ← next node on path P_i;
15.                       v ← d;
16.   compute P_e = K_s! × ∏_{i=1}^{K_s} ( C_{P_i} / (E − i + 1) );
end
```

Figure 13.   Pseudo code for Pe computation

This computation is exact for graphs where alternative paths for edge-disjoint paths do not exist. However, in networks that are at least partially meshed, this will rarely be the case and this is why P_e is an approximation for the probability P_r.

The P_e model as a heuristic was tested for graphs that have less than 12 nodes and results were verified in [CHU03]. The P_e model appeared to be more accurate than other heuristics developed before it, having an accuracy of almost 92% in determining the best node in a network to host a critical server.

B.   ASSUMPTIONS OF THE P_E MODEL

The heuristic model, P_e, introduced above, works under two assumptions:

1.   The failures have uniform distribution.

2.   Nodes have independent failures.

This means that the computation of the P_e is exact if you have independent and uniformly distributed failures, which is rarely the case. The validity of these assumptions is considered below.

1.   Validation of the Underlying Assumptions

In Chapter III, two criteria were given to compare and rank the nodes of a network. These were:

1.   Network survivability metric based on the edge-connectivity factor (K_e).

2.   Probability of failure of a link given K_e number of edge failures.

Given K_e number of failures, one can always rank the nodes in a network as discussed in Chapter III Section C. However, it was shown that $P_r\{\mathrm{cut}(s_1, d) = 1 \mid K_e \, \mathrm{edge\,failures}\}$ may not predict $P_r\{\mathrm{cut}(s_1, d) = 1 \mid (K_e + 1) \, \mathrm{edge\,failures}\}$. The example used is shown in Figure 14. In this example, it was determined that ranking among nodes does not stay the same if the number of failures exceeds K_e.
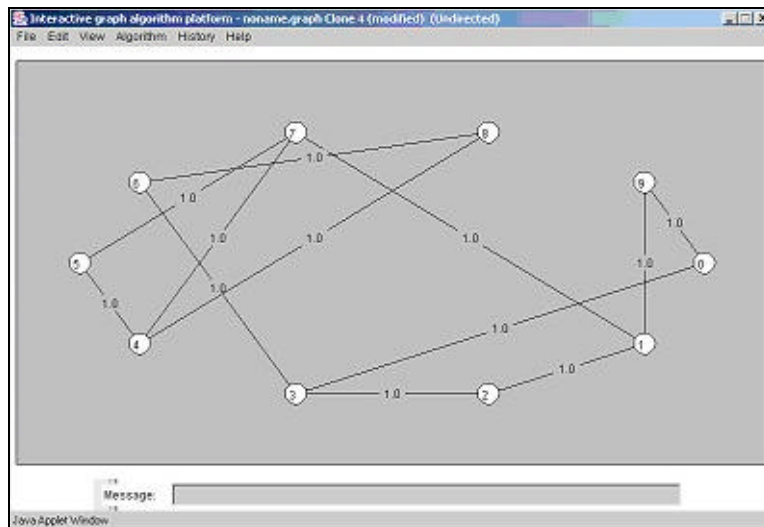
Figure 14.   The graph used to verify Pe model

In Figure 15, s_1 (source node), selected as Node 6, is to be compared to s_2, which is Node 7. (See Figure 17 for s_2) The two selected nodes have the same number of K_e, equal to two. Therefore, the second criteria must be evaluated, requiring that the minimum number of cut sets, min-cuts, given K_e failures, be computed. Execution of the simulation program determined that s_1 has 5 min-cuts and s_2 has 5 min-cuts. Therefore, according to P_e model, one would think that the two nodes would be ranked the same. However, if the number of failures exceeds K_e, three in this example, the simulation program determined that s_1 has 58 cut sets and s_2 has 62 cut sets, which means the ranking of the nodes should not be the same.

Figure 15. $s_1 = 6$ (source), t =0 (Destination)
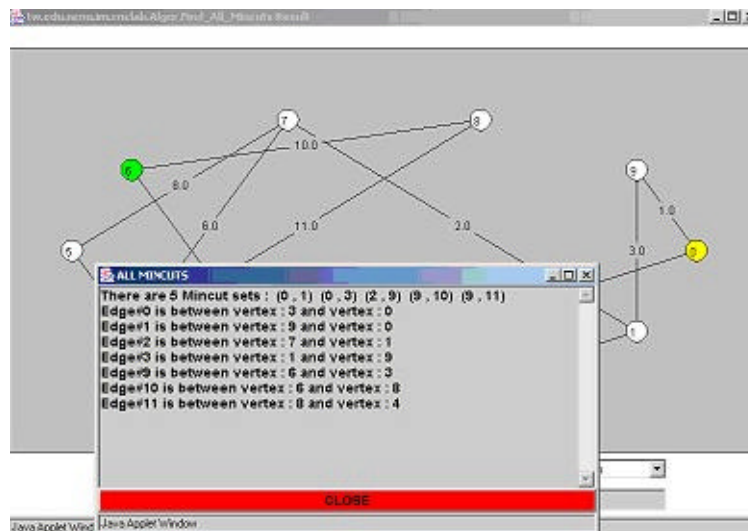
Figure 16.   Shows the min-cut computation for s_1

28
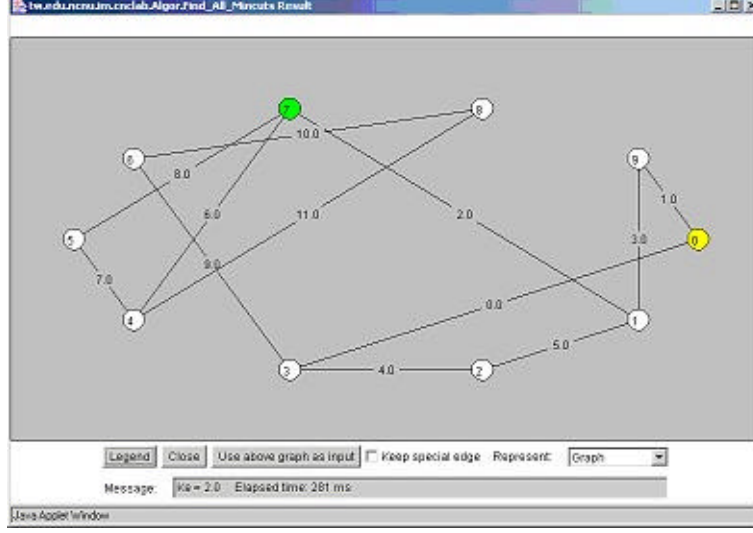
Figure 17. $s_2 = 7$ (source) and t=0 (destination)
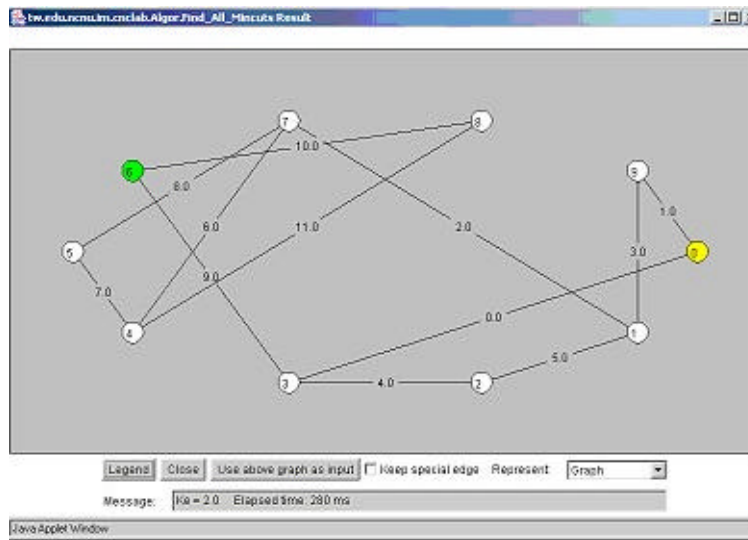
As a result, a connectivity-based survivability metric, developed in [AKT03] was used and the heuristic developed in [CHU03] was evaluated. This heuristic is used to compare the connection reliability of two nodes to a common destination node when these two nodes have the same number of edge-disjoint paths to that destination. It is based on estimating the probability of each of the nodes being cut from the server given same number of link failures. The topology in Figure 14 was used to validate and refine this heuristic.

The simulation results show that the statement,

If $P_r\{cut(s_1, d) = 1 | K_e \text{ edgefailures}\} < P_r\{cut(s_2, d) = 1 | K_e \text{ edgefailures}\}$

then $P_r\{cut(s_1, d) = 1 | n \text{ edgefailures}\} < P_r\{cut(s_2, d) = 1 | n \text{ edgefailures}\}$

where $n > K_e$, may not be always true. This result implies that, while using the P_e model, the ranking of the nodes may not stay the same, given that the number of failures is more than K_e.

THIS PAGE INTENTIONALLY LEFT BLANK

V.   A REALISTIC APPROACH FOR NETWORK SURVIVABILITY

A.   LIMITATIONS OF THE EXISTING FAILURE MODELS

The survivability computations presented in the previous chapters, from a mathematical point of view, assume that:

1.   All failures are equally likely.

2.   The failures are mutually independent.

These assumptions do not adequately reflect the nature of real world network environments. Typically, different nodes or links can have different failure probabilities. More important, real systems show correlated failures. Correlated faults can result in reduced system reliability and availability [http://oceastore.cs.berkeley.edu, September 2003].

Server failures may be correlated because they share network routers, software bugs, configuration problems, operating systems, etc. Failure independence of the nodes in a network may be searched and a set of independent nodes may be modeled. Then selective use of resources from among these independent sets can be implemented to fight against correlation of failures. This can be achieved by grouping highly correlated nodes together and consider each group a single domain. The means to model and compute the probability of failure of individual correlated domains remains unsolved because of complex conditional probability computations involved. Therefore, our focus is relaxing Assumption 2 above by computing all failure probabilities within the network while including correlated failures. This is likely a more realistic metric for network administrators. The next section explains the algorithm used to compute the probability of failures for correlated components.

B.   COMPUTING CORRELATED COMPONENT FAILURES

In a graph, one first needs to know the entire node-disjoint cut-sets that disconnect paths between s and d. Given the probability of failures, one must then compute the probability of s-d being cut by attacking the problem in a brute-force manner, since there

is not a way of modeling dependent node failures as independent failures. If all the failures can be viewed as independent failures, then computing $P_r\{\text{cut(s,d)} = 1\} = |K_e \text{ edge failure}\}$ would be trivial by simply multiplying the probability values given.

The heart of the algorithm is to compute the probability of all the node cut-sets. Without loss of generality, let us assume there are a total of four cut-sets. Consider the recursion given in Equation (3) below:

$$P(A \cup B \cup C \cup D) = P(A) + P(B \cup C \cup D) - P(A \cap (B \cup C \cup D)) \qquad (3)$$

where A, B, C, and D represent the node cut sets.

The equation above is a result of the well-established Equation:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \qquad (4)$$

Equation (3) can be solved recursively by the following equation:

$$P(A \cup B \cup C \cup D) = P(A) + P(B \cup C \cup D) - P((AB) \cup (AC) \cup (AD)) \qquad (5)$$

where AB, AC or AD may include correlated component failures.

A basic Java implementation of this approach can be found in the Appendix. In the implementation node cut-sets that disconnect the source node and destination node are stored in an array of vectors, depicted as "N [ ]" in Figure 18. Independent node failures, dependent node failures, and their corresponding values are stored in a hash table. The recursive computation given in Equation (5) is implemented in the procedure called findProbability (N [ ]). The pseudo-code for this algorithm is given in Figure 18.

```
double findProbability((Vector N[ ])
{
        //base case
        if (N.length = = 1)

                double p1=1.0
                if (failures of some nodes of N[0] are correlated)
                    double prob  ◄—   get the joint failure probability of corresponding
                                    nodes from the probability table

                    N4[ ]  ◄—   N[0] – {correlated nodes}
                    return  p1 * prob * findProbability(N4)


                else    //In this case : failures of all nodes are independent
                        //then simply get the node probabilities and multiply them
                         all
                    for (each node in N[0])
                        prob  ◄—  get failure probability of the node from the
                                    probability table

                        p1  ◄—  p1* prob
                    return p1

        else    // now N.length > 1
                //initialize vectors for recursive computation

                Vector N1[ ]
                Vector N2[ ]
                Vector N3[ ]

                N1 = N[0] //only the first vector
                N2 = { N[1], N[2], ……, N[n] } //removal of the first vector
                N3 = { N[0].N[1], N[0].N[2],…., N[0].N[n] } //cross product of N1 and
N2

                Return

                    findProbability(N1) + findProbability(N2) + findProbability(N3)
```

Figure 18.   findProbability procedure pseudo code

33

A basic topology example used to run and verify the results of the program is given in Figure 19.
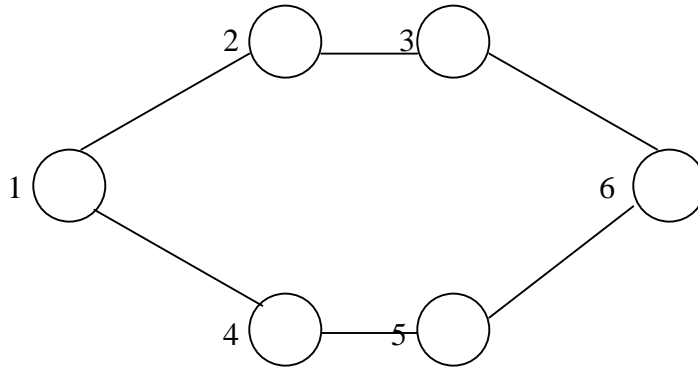
Figure 19.   Example topology to verify java implementation of algorithm

In this example, Node 1 is the source node and Node 6 is the destination node. Nodes 2 and 3 are dependent nodes, as are Nodes 4 and 5. This means if Node 2 fails it increases the probability of the failure of Node 3. Node cut sets of this example are: { [2,5], [2,4], [3,4], [3,5] }.

The following probability values given:

P(2|3) = 0.3 (this is the conditional probability value because 2 and 3 are dependent nodes)
P(4|5) = 0.5
P(2) = 0.4
P(3) = 0.3
P(4) = 0.4
P(5) = 0.4

The result for this graph P(1 and 6 being cut) = 0.12 is calculated by hand computation and verified by the Java simulation given in the Appendix.

VI. CONCLUSIONS AND FUTURE WORK

A.   SYNOPSIS AND CONCLUSIONS

When there is an attack to a DON/DOD network, critical data servers should be relocated based on the current situation. The solution developed by this thesis gives the decision makers criteria that will help them relocate the servers to parts of the network where the services are more survivable.   This thesis explored this solution in the following manner.

First, to compare and rank the nodes of a network, two criteria were used:

1.   Network survivability metric based on the edge-connectivity factor (K_e).

2.   Probability of failure of a link given K_e number of edge failures.

Second, an algorithm to enumerate all min-cuts and near min-cuts was introduced. The implementations of these approaches were evaluated and tested using various graphs. However, because of the complexity issues involved in maximum flow, min-cut algorithms, these algorithms were practical only for certain types of networks.

Third, a heuristic model, P_e, based on edge-connectivity, (K_e), was explained. The P_e model had previously been tested for graphs that have less than 12 nodes and the results were verified in [CHU03]. During this research it was shown that the assumption,

If $P_r\{cut(s_1,d)=1|\ K_e\ edgefailures\}\ <\ \quad P_r\{cut(s_2,d)=1|\ K_e\ edgefailures\}$

then $P_r\{cut(s_1,d)=1|\ n\ edgefailures\}\quad <\ P_r\{cut(s_2,d)=1|\ n\ edgefailures\}$

(For n =1, 2, 3....where n> K_e), may not always hold.

Finally, the limitations of the current survivability metrics were discussed. The author proposed a way to relax assumptions of the failure models. Two Java-based programs were used to simulate the effects of node failures. The simulation results are the computation of the connectivity factor of the nodes and the number of minimum cut-sets of the sample network. The implementation platform for the survivability metrics was a prototype, called the Server and Agent-based Active Network Management (SAAM) system, which was proposed by Prof. Geoffrey Xie in 1998 and developed by graduate

35

students of NPS over the ensuing years. A Java simulation was written by the author to verify the viability of the approach taken and to compute $P_r\{\text{cut(s,d)} = 1 | K_e \text{ edge failures}\}$. The time frame of this research did not allow for pertinent statistics to be gathered.

B. FUTURE WORK

Given current engineering practices, developing models that are reasonably independent of the details of failure nodes, probabilities and correlations is difficult at best. However, modeling dependent node failures in a way such that they can be represented as independent node failures will ease the complexity of the computation of $P_r\{\text{cut(s,d)} = 1 | K_e \text{ edge failures}\}$. Methods for implementing such models should be investigated

Metrics, other than K_e, need to be established to quantify the fault tolerance, safety, reliability, and performance of the nodes in a network.

Finally, more work needs to be done to verify and implement the proposed algorithm for large-scale networks.

APPENDIX. JAVA SIMULATION CODE

```java
import java.util.*;
import java.awt.*;
import java.awt.event.*;


import javax.swing.*;



/**
 * Title: Computer (s,t) failure probability
 * Description: Brute-force computation and working with correlated node
failures.
 *  1. Link failures have been converted into node failures by transforming the
graph.
 * 2. Correlated node failures modeled as joint failure probabilities.
 *
 * Copyright:    Copyright (c) 2003
 * Company:
 * @author Ozlem Ozkok & Geoffrey Xie
 * @version 1.0
 */

public class ComputeProbability {


  private final int NUM_CUT_SETS = 4;
  private Vector nodeCutSets[] = new Vector[NUM_CUT_SETS];
  private double p;
  private Hashtable probTable;


  public ComputeProbability()
```

```
{
 for (int i = 0; i < NUM_CUT_SETS; i++)
 {
  nodeCutSets[i] = new Vector();
 }

 //Known node cutsets
 //Note: The cutset are not mutually exclusive.
 nodeCutSets[0].add(new Integer(2));
 nodeCutSets[0].add(new Integer(5));

 nodeCutSets[1].add(new Integer(2));
 nodeCutSets[1].add(new Integer(4));

 nodeCutSets[2].add(new Integer(3));
 nodeCutSets[2].add(new Integer(4));

 nodeCutSets[3].add(new Integer(3));
 nodeCutSets[3].add(new Integer(5));


 //Known failure probabilities
 //Note: Failures of node 2 and 3 are correlated and so are node 4 and 5.
 probTable = new Hashtable();

 probTable.put("2.3", new Double(0.3));
 probTable.put("4.5", new Double(0.5));
 probTable.put("2", new Double(0.4));
 probTable.put("3", new Double(0.3));
 probTable.put("4", new Double(0.4));
 probTable.put("5", new Double(0.4));
```

```java
      System.out.println(probTable);

    p = findProbability(nodeCutSets);

  }//end constructor



  private double findProbability(Vector N[])
  {
   if (N.length == 1)
   {
     System.out.println("Inside the base case " + N.toString());
     double p1 = 1.0;

     int numNodes = N[0].size();

     for (int nodeCount = numNodes; nodeCount > 1; nodeCount--)
     {
       //Now search the probTable; starting with longest key
       for (Enumeration e = probTable.keys(); e.hasMoreElements();)
       {
         String key = (String) e.nextElement();
         if (key.length() == nodeCount * 2 - 1)  //account for "."
         {
           //Convert key e.g., "1.2.3" into vector {Integer(1),Integer(2),Integer(3)}
           Vector keyVector = convertToVector(key);

           //check for match
           boolean matching = false;
           System.out.print("Key Vector = " + keyVector + "; N[0] = " + N[0] +
"\n");
```

```
                    for (int j = 0; j < keyVector.size(); j++)
                    {
                      matching = false;
                      for (int k = 0; k < N[0].size(); k++)
                      {
                        if ((keyVector.get(j)).equals(N[0].get(k)))
                        {
                          matching = true;
                          break;
                        }
                      }
                      if (!matching)
                        break;
                    }

                    if (matching)
                    {
                      if (!probTable.containsKey(key))
                      {
                        System.out.println("Error: Required probability variable not given for
node sequence " + key);
                      }
                      else
                      {
                        Double prob = (Double) probTable.get(key);


                        //Remove corresponding nodes from N[0] since their joint probability is
found.
                        //Note: their joint failure is independent of failures of the other N[0]
nodes


                        Vector N4[] = new Vector[1];
                        N4 = formN4(N, keyVector);
```

40

```java
              return (p1 * prob.doubleValue() * findProbability(N4));
           }


        }
       else
       {
          System.out.println("No match for key: " + key + "\n");
       }


     }//end of if (key.length == ...


  }
}


//Now nodeCount is 1 and in this case, failures of all nodes are independent.
//So just multiply their failure probabilities together
System.out.println("Now failures of all nodes in N[0] are independent.\n");


for (int i = 0; i < N[0].size(); i++)
{
  String key = "" + N[0].get(i);


  if (!probTable.containsKey(key))
  {
     System.out.println("Error: Required probability variable not given for node
" + N[0].get(i));
     System.exit (1);
  }
  else
  {
    Double prob = (Double) probTable.get(key);
    p1 = p1 * prob.doubleValue();
```

```java
      }
     }

    return p1;


  }
  else
  {
   Vector N1[] = new Vector[1];
   Vector N2[] = new Vector[N.length - 1];
   Vector N3[] = new Vector[N.length - 1];


   N1 = formN1(N); // subset with only the first vector
   N2 = formN2(N); // after removal of first vector
   N3 = formN3(N); // cross product of N1 and N2


   return findProbability(N1) + findProbability(N2) - findProbability(N3);


  }
}//end method findProbability



private Vector[] formN1(Vector temp[])
{
 Vector N1[] = new Vector[1];
 N1[0] = temp[0];
 return N1;
}

private Vector[] formN2(Vector temp[])
{
 Vector N2[] = new Vector[temp.length-1];
```
42

```java
    for (int c = 1; c < temp.length; c++)
    {
      N2[c-1] = temp[c];
    }


    return N2;
  }


  private Vector[] formN3(Vector temp[])
  {
    Vector N3[] = new Vector[temp.length-1];


    for (int c = 0 ; c < temp.length - 1; c++)
    {
      N3[c] = new Vector();
    }


    for (int i = 1; i < temp.length; i++)
    {
      //Merge elements of temp[0] and temp[i] and put them into N3[i-1]
      for (int j = 0 ; j < temp[0].size(); j++)
      {
        N3[i-1].add(temp[0].get(j));
      }


      for (int k = 0; k < temp[i].size(); k++)
      { //no duplicate is added
        if (!N3[i-1].contains(temp[i].get(k)))
        {
          N3[i-1].add(temp[i].get(k));
        }
```

```java
    }


  }

  return N3;
}



private Vector [] formN4(Vector temp[], Vector removeVector)
{
  Vector N4[] = new Vector[1];
  N4[0] = new Vector();


  for (int count = 0 ; count < temp[0].size(); count++)
  {


    Integer temporary = (Integer) temp[0].get(count);
    System.out.println("N4[0] = " + N4[0] + "; temporary = " + temporary);


    if (!removeVector.contains(temporary))
    {
      N4[0].add(temporary);
    }
  }
  return N4;
}



private Vector convertToVector(String key)
{
  Vector v = new Vector();


  StringTokenizer token = new StringTokenizer(key,".");
```
44

```java
    while (token.hasMoreTokens())
    {
     Integer id = Integer.decode(token.nextToken());
     v.add(id);
    }


   System.out.println("Key = " + key + " ; new vector = " + v.toString());
   return v;
 }


 public String toString()
 {
  return "(s, t) Cut Probability = " + p;
 }

 public static void main( String args[] )
 {
  ComputeProbability test = new ComputeProbability();
  System.out.println(test.toString());

 }//end main
}//end ComputeProbability
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[AKT03]   Baris Aktop, *A Framework for Maximizing Survivability of Network Dependent Services*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 2003

[CHU03]   Eng Hong Chua, *Determine Network Survivability Using Heuristic Models*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 2003

[CEB98]   Arthur K. Cebrowski, John J. Garstka, *Network Centric Warfare :Its Origin and Future,* U.S. Naval Institute Proceedings, 1998

[DES02]   Paul   Desmond,   *Experts   Warn   of   Cyber   Attacks,* www.itmanagement.eathweb.com, March 2002

[DIE01]   S. Dietrich and PYA Ryan, *The Survivability of Survivability,* Software Engineering Institute Carnegie Melon University, September 2001

[ELL99]   Robert J. Ellison, David A. Fisher, Richard C. Linger, Howard F. Lipson, Thomas A. Longstaff, Nancy R. Mead, *Survivability: Protecting Your Critical Systems,* CERT Coordination Center Software Engineering Institute Carnegie Melon University, 1999

[FIS99]   David A. Fisher, Howard F. Lipson, *Emergent Algorithms: A New Method for Enhancing Survivability in Unbounded Systems,* Software Engineering Institute Carnegie Melon University, 1999

[JHA00]   Jha, S. and Wing, J. M., "*Survivability Analysis of A Networked System*", Proceedings of the 23rd International Conference of Software Engineering, pp. 307-317, July 2001

[LAW00]   Cliff Lawson, *TID*, The Weaponer, December 2000

[MEA00]   Mead N. R. *et al.*, "Survivable Network Analysis Method", Technical Report, Carnegie Mellon Software Engineering Institute, CMU/SEI-2000-TR-013, September 2000

[SHI01]   Timothy Shimall, *Countering Cyber War,* NATO Review, Winter 2001

[SUL99]   Sullivan, K. *et al.*, "Information Survivability Control Systems", *Proceedings of 1999 International Conference of Software Engineering* (ICSE 99), pp. 184-192, Los Alamitos, May 1999

[UMA01]   Umar, A. *et al.*, "Intrusion Tolerant Middleware", *Proceedings of DARPA Information Survivability Conference and Exposition* (DISCEX 01) vol. 2, 2001

[WEL00]   Wells, D. *et al.*, "Software Survivability", *Proceedings of DARPA Information Survivability Conference and Exposition* (DISCEX 01) vol. 2, 2000

[WOO00]   R. Kevin Wood, Ahmet Balcioglu, *Enumerating Near-Min S-T Cuts in Network Interdiction and Stochastic Integer Programming*, ed. Woodruff, D.L., Kluwer Academic Publishers, 2003, pp. 21-49.

[XIE98]   Geoffrey G. Xie, *SAAM: An Integrated Network Architecture for Integrated Services,* IEEE, 1998

[XIE02]   Geoffrey G. Xie, *CY03 Homeland Security Research & Technology Research Proposal: "Critical Infrastructure Protection: Maximize Survivability of a Network Dependent Service,"* 2002

[YUR99]   William Yurcik, *Adaptive Multi-Layer Network Survivability: A Unified Framework for Countering Cyber Terrorism,* Telecommunications Program University of Pittsburgh, 1999

INITIAL DISTRIBUTION LIST

1.   Defense Technical Information Center
        Ft. Belvoir, Virginia

2.   Dudley Knox Library
        Naval Postgraduate School
        Monterey, California

3.   Professor Geoffrey Xie
        Department of Computer Science
        Naval Postgraduate School
        Monterey, California

4.   Professor Alex Bordetsky
        Department of Information Sciences
        Naval Postgraduate School
        Monterey, California

5.   Deniz Kuvvetleri Komutanligi
        Kutuphane
        Bakanliklar, Ankara, TURKEY

6.   Deniz Harp Okulu Komutanligi
        Kutuphane
        Tuzla, Istanbul, TURKEY

7.   Bilkent Universitesi Kutuphanesi
        Bilkent, Ankara, TURKEY

8.   Orta Dogu Teknik Universitesi Kutuphanesi
        Balgat, Ankara, TURKEY

9.   Bogazici Universitesi Kutuphanesi
        Bebek, Istanbul, TURKEY